Algorithmi: software system to support the learning of programming

António Manso
Department of Information and Communication Technologies
Polytechnic Institute of Tomar
Tomar, Portugal
manso@ipt.pt

Célio Gonçalo Marques
Department of Information and Communication Technologies
Polytechnic Institute of Tomar
Tomar, Portugal
celiomarques@ipt.pt

Paulo Santos
Department of Information and Communication Technologies
Polytechnic Institute of Tomar
Tomar, Portugal
psantos@ipt.pt

Learning programming is a hard frustrating process for students with greatest difficulties. Programming involves thinking abstractly, which requires motivation and commitment. For this reason, traditional teaching approaches are being replaced by more attractive technological solutions such as Logo and Scratch. Although learning how to program with these tools is extremely fun and easy, they are very different from the commercial programming languages students will come across in the working world. This led to the creation of Algorithmi, a system designed to support the learning of algorithmics composed by an algorithmic language that can be translated into traditional programming languages, an algorithm editor that minimises mistakes and corrects algorithms; and an information system with a database of exercises which can be fed-in by the instructor. Algorithmi is being developed since 2016 and its usability was tested by faculty members and students. The enthusiasm was unanimous and, in the 2018/2019 academic year, Algorithmi will start to be taught in all introductory programming courses offered at IPT, and consequently its assessment in learning context.

Keywords — Algorithmi, Learning, Information System, Programming.

I. Introduction

Introductory programming courses have a high drop-out rate, which hinders progression and programme completion [1] [2] [3] [4]. As programming is a basic skill required in several areas of computer engineering, failure hampers the learning of other related content. Learning how to program is not an easy task; it requires the combination of personal and cognitive skills that students must develop. Personal skills include proactivity, persistence, confidence and responsibility. Cognitive skills include analogy, logical reasoning and abstract thought required to solve computer problems. However, these skills are not enough. Students must be able to hierarchize them and use them simultaneously [5].

This problem is cross-cutting to all countries, institutions and teaching levels [6]. To overcome this problem, a wide range of tools from integrated development environments to learning micro-worlds have been created [6]. Although they are an important contribution to teaching and learning of programming, none of these tools has proved effective [7], which led to the development of Algorithmi in order to reduce the failure rate of IPT students in programming courses.

II. Environments developed to support teaching and learning of programming

There are several platforms that share the same objectives of Algorithmi such as Portugol IDE [8], the predecessor of Algorithmi, and various Brazilian Portuguese platforms such as VisualAlg [9], Portugol Studio [10] or G-Portugol [11]. All of them, however, make use of a textual programming language.

When using Portugol IDE [8] to teach algorithmics it was observed that in a textual programming language, even in Portuguese, students have had some trouble with encoding the algorithm.

Thus we analysed new tools used to teach algorithms that represented a shift in the classical paradigm for writing algorithms resorting to a textual language such as Logo, Scratch, Code.org and Portugol IDE.

A. Logo

Logo [12] was one of the first programming languages designed to teach algorithms to children. The language has a set of controls that allow to move a turtle in an interactive way. The limited set of Logo controls allows to rapidly learn the language and move forward to learning and experimenting with algorithms. Although Logo still uses a textual language to implement the algorithms, it is equipped with a visual execution module.

B. Scratch

Scratch [13] is a tool to program games, animations and interactive stories developed by Lifelong Kindergarten Group at the MIT Media lab. Although it was originally designed for 8-16 year olds, it is now used by students of all ages. Programming is graphic and makes use of blocks that fit together to build an algorithm, that is why there is no need to learn a programming language to learn algorithms.

C. Code.org

Code.org [14] is a Web portal designed by Google, Microsoft, Facebook and Twitter with the aim of sharing content, courses and classes for algorithm learning. Like Scratch, Code Studio uses a block-based programming language in which students are confronted with challenges involving fictional worlds such as Star Wars or Minecraft.

D. Portugol IDE

Besides allowing the construction of algorithms in textual language, Portugol IDE [8], was already equipped with a module for the creation of flowcharts. Flowcharts have interesting learning features such as a very limited set of instructions and simple syntax. However, total freedom between blocks was allowed, enabling the creation of wrong flowcharts or flowcharts which, while functionally correct, could not be translated into structured programming languages.

978-1-5386-6505-3/18/$31.00 ©2018 IEEE

Although these programming environments are easy to learn, the problems involved are outside the scope of programming and transition to traditional programming languages is not immediate. There was therefore the need to create a system that would meet algorithms and computation learning needs and facilitate the transition to traditional programming languages. Algorithmi.

Algorithmi is a programming learning system that arose from the need to further develop Portugol language including new features such as manipulation of homogeneous data structures and function introduction. These two features allow to expand its usage to different content of programming courses/modules and to develop more complex algorithms.

Algorithmi consists of three models that interact together: The algorithmic language which allows algorithm specification; the editor which allows its encoding and execution, and the information system that supports learning.

III. Algorithmic language

The algorithmic language is a further development of Portugol whose main feature is to allow students to write algorithms in their mother tongue. Traditionally, programming learning is done through languages with English lexicon and a rigid syntax in order to build a safe and sound software. However, these languages are not appropriate for teaching the basic concepts of programming.

The difficulties felt by students with poor English when using traditional programming languages led us to rewrite the execution core of Portugol to support different languages other than Portuguese. This feature allows to include other languages, even those that do not use western alphabet such as Arabian, Chinese or Greek.

Algorithmic language is the formal language used to write algorithms. It is based on symbols translated into the mother tongue of students. This feature allows an algorithm written in Portuguese to be visualised and executed for example in French, Greek, Arabian or Russian. This translation and execution (described further in the text) is done by the editor.

A. Data types

Table 1 presents simple data types of the algorithmic language used in Portugol. The void type was introduced to allow the definition of procedures that do not calculate values. Integer and real types allow to represent numeric data, the logic type represents true and false values and the text type allows the representation of a set of characters. In the process of simplifying language, we have eliminated the character type as it can be represented by the text type.

B. Computational expressions

For data manipulation arithmetic, logic and relational operators have been implemented. Table 2 shows the operators and its encoding in Portugol. Operator symbols are parametrizable, which allows to change the way how computational expressions are displayed.

Besides operators in Table 2, the calculation functions defined are those that traditionally belong to programming languages. The terms used in these functions can be parametrized in the mother tongue of students.

TABLE 1 – DATA TYPES

Symbol	Portugol	Description
VOID	vazio	used in functions that return nothing (void)
INTEGER	inteiro	integral numbers
REAL	real	Real numbers
TEXT	texto	Set of characters
LOGIC	lógico	logic values

TABLE 2 - ARITHMETIC, RELATIONAL AND LOGIC OPERATORS

Symbol	Portugol	Description
SET	=	Atribuition
SUM	+	Sum
SUB	-	Subtraction
MULT	*	Multiplication
DIV	/	Division
MOD	%	Rest of Division
POWER	^	Power
EQUAL	==	Equality
DIFFERENT	!=	Difference
GREATHER_THAN	>	Bigger than
GREATHER_OR_EQUAL	>=	Bigger or equal
LESS_THAN	<	Less than
LESS_OR_EQUAL	<=	Less or equal
NOT	!	Not
AND	&&	And
OR	||	Or

C. *Instructions*

Table 3 shows the instructions defined in the language.

TABLE 3 – CONTROL STRUCTURES

Symbol	Portugol	Description
COMMENTS	//	Coments
BEGIN	inicio	Start of main program
END	fim	End of block
DEFINE	definir	Defines stored variables
READ	ler	Reads the value of a variable from the console
EXECUTE	executar	Executes calculations or functions
WRITE	escrever	Writes information on the console
IF ELSE	se senão	Executes decisions
WHILE	enquanto	Executes a set of instructions with initial control
DO	faz	Executes a set of instructions with final control
ITERATE	iterar	Executes a set of instructions a pre-determined number of times
JUMP	saltar	Interrupts or continues a cycle
FUNCTION	função	Defines a function
RETURN	retornar	Returns one value from a function

- COMMENTS - Allows to define a comment that helps understand the algorithm. The instruction allows to only comment the lines, which means that there are no comments during instructions.
- BEGIN – Defines the start of the main program Serves as the entry for algorithm execution.
- END – Defines the end of a block: function, decision or cycle.
- DEFINE – Allows to define a variable of a given type which is initialized with a value. The instruction can also be used to define data vectors. Vectors can be multidimensional and there is no limit to the number of dimensions.
- READ – Allows to read variables from the console. The instruction syntax allows to read a pre-defined variable or to read and define the variable simultaneously.
- EXECUTE – Executes calculations with computational expressions. These expressions can have functions defined by the user. It also serves to execute procedures.
- WRITE – Writes to the console the result of a computational expression.
- IF ELSE – Executes flow diversion through a decision.
- WHILE – Executes an initial-condition controlled cycle.
- DO – Executes a final-condition controlled cycle.
- ITERATE – Iterates a numeric set in which start, end and pace are defined.
- JUMP – Changes a cycle flow by interrupting it or continuing it to the condition.
- FUNCTION – Defines a function. Functions may be iterative or recursive.
- RETURN – Returns from a function.

Table 4 shows the "Hello World" program in algorithmic language and its translation to Portuguese and Greek.

TABLE 4 – "HELLO WORLD" PROGRAM IN ALGORITHMIC LANGUAGE, PORTUGOL AND GREEK.

```
BEGIN MAIN_PROGRAM_NAME
   WRITE "Hello World"
END MAIN_PROGRAM_NAME
```

```
inicio ProgramaPrincipal
    escrever "Hello World"
fim ProgramaPrincipal
```

```
αρχίζουν Κύριοπρόγραμμα
    γράφω "Hello World"
τέλος Κύριοπρόγραμμα
```

IV. ALGORITHM EDITOR

An algorithmic language can be encoded and executed directly in text form, however, this is an impractical way of editing algorithms and ineffective for teaching and learning.

For teaching and learning we have developed an editor to explore algorithmic language with a simple and attractive graphic interface. The editor allows the algorithm to be edited in text format in the mother tongue of the student or using flowcharts. Accumulated experience with the Portugol language to teach programming in the several degree programmes offered at IPT shows that flowcharts are most suitable to learn basic programming and advanced text editing.

A. *Flowchart Editor*

The flowchart editor was designed to minimise student errors when editing an algorithm. Every algorithm starts with an execution line and can be replaced by instructions that generate new execution lines. When students eliminate instructions flow lines are updated. Figure 1 shows the menu

that allows the user to replace a flow line (green arrow) by an instruction.

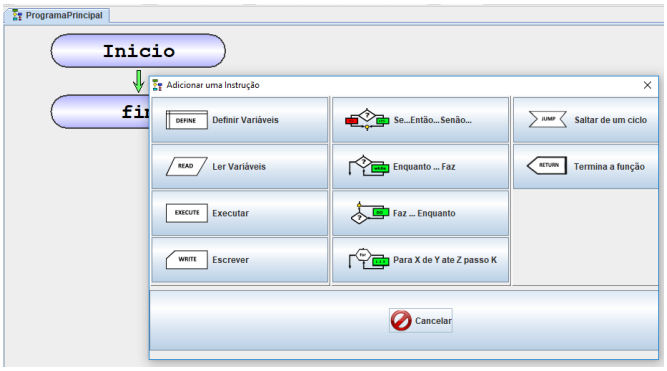


Fig. 1 – Instruction insertion

Figure 2 shows the variable setting wizard. The example shows the setting of a 2x3 integer matrix. The instructions inserted by students can be written in their mother tongue (green area) or through wizards that help them create the instructions (red area).

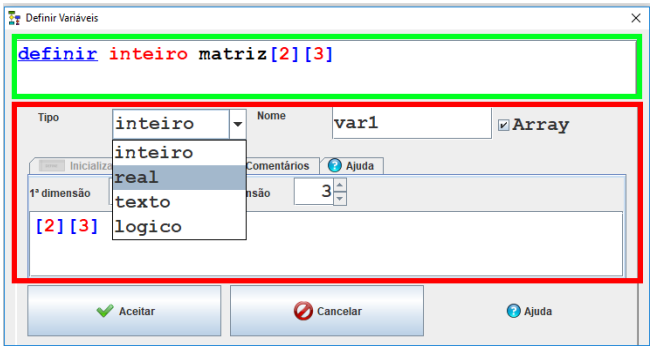


Fig. 2 – Instruction insertion

Every instruction allows to write comments which can later be visualized in the algorithm (figure 3) through context information.

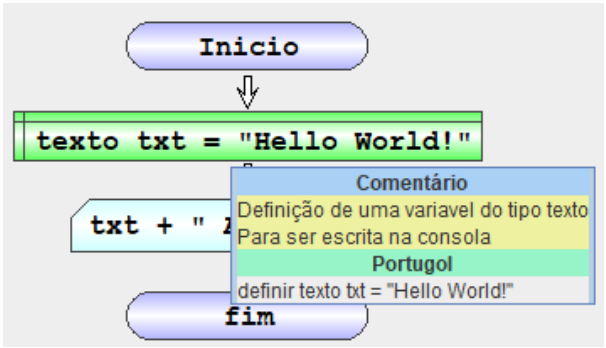


Fig. 3 - Visualisation of algorithm comments.

Every instruction has a wizard that helps the student to write the instructions correctly. The flow line only accepts instructions if they are syntactically correct, thereby ensuring that the algorithm will always be compilable.

When the student eliminates an instruction such as, for example, the definition of a variable, the algorithm becomes incorrect and he/she perceives it because the colour of the instruction changes and context information appears on the display (figure 4).

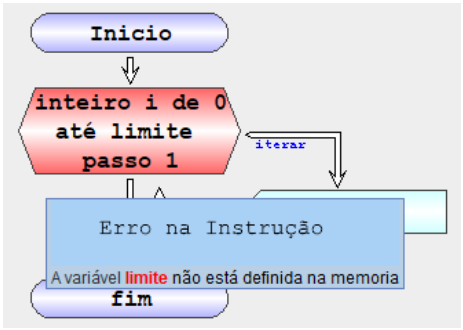


Fig. 4 – Visualization of instruction errors.

B. *Algorithm Executor*

When the algorithm is finished it may rapidly be executed to check if it is correct (Figure 5). The execution of the algorithm allows the user to visualise algorithm inputs and outputs and provides some information about their execution: time, instructions and calculations. This information is important to learn the complexity of algorithms and the computational effort required to execute them.

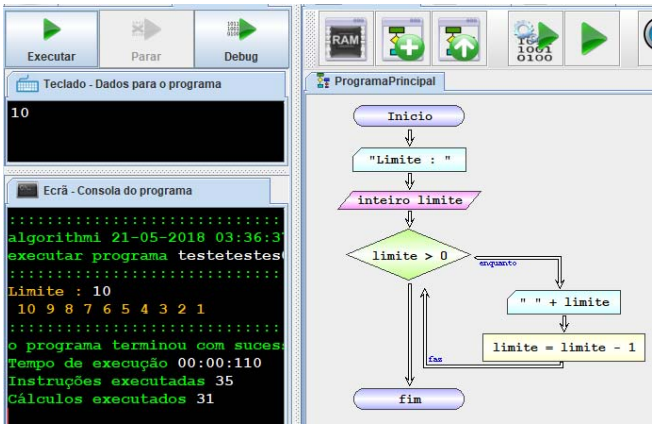


Fig. 5 – Algorithm execution

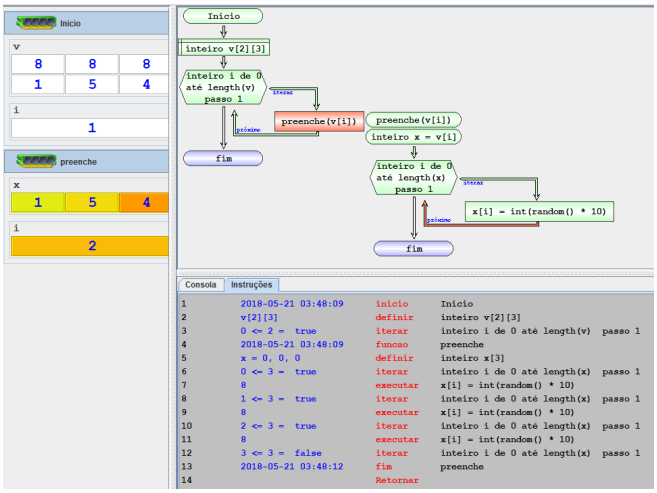


Fig. 6 – Algorithm debugging

Algorithms may also be debugged through step-by-step execution. Figure 6 shows an algorithm that fills up the 2x3 matrix through a function that receives a vector as a parameter.

In this mode, it is possible to graphically display the variables stored for every function and the values that are being altered by the instructions. It is also possible to see the instructions that are being executed and the history of already executed instructions. It is also possible to see the functions and parameters graphically.

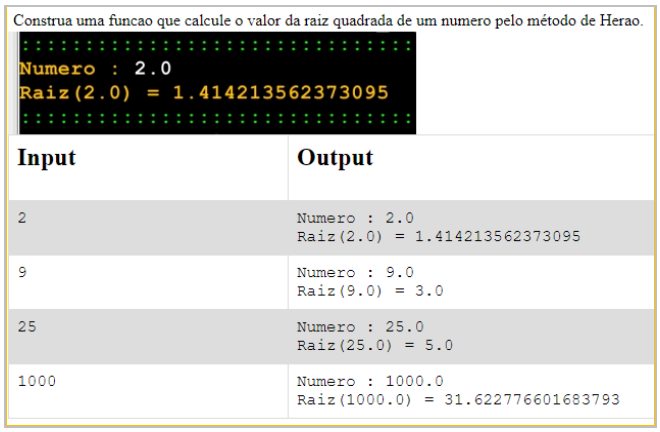
V. Information System

Algorithmi is a learning environment that makes use of an information system to help students to learn programming and instructors to assess learning outcomes.

The system consists of a server that provides services to the algorithm editor, and programming exercises for students as well as information on learning progression.

A. *Programming exercises*

Algorithmi is comprised of a database containing problems (figure 7) that guide student in their learning of a wide range of algorithms (Table 4) from simple to multidimensional array algorithms with recursive functions.



Input	Output
2	Numero : 2.0 Raiz(2.0) = 1.414213562373095
9	Numero : 9.0 Raiz(9.0) = 3.0
25	Numero : 25.0 Raiz(25.0) = 5.0
1000	Numero : 1000.0 Raiz(1000.0) = 31.622776601683793

Fig. 7 - Programming Exercises

Every exercise has an introduction that may include a picture and a series of inputs and outputs that make it correctable (figure 8). The system compares the inputs produced by the student's algorithm with expected inputs, informs the student of the mistakes made and assigns a score to it. Algorithm assessment compares text and numeric values in the output in order to detect format and round-off errors, and real values. Format (figure 8) and round-off errors receive small penalties while absence of text or values have larger penalties. The presence of unexpected values and text is also subject to penalties.

The fact that exercises are self-correctable allows students to assess their learning progress. On the other hand, it allows the instructors to make more exercises available and provide support to students in need. Once an exercise is solved, students submit it through the system, receive a score and earn points in their personal account. Thus students receive a reward for their effort and the motivation for solving additional exercises.

For each exercise the system translates the algorithm into traditional programming languages (figure 9) such as C, C++, Java, Python and others.

The translation of algorithms into a high-level programming language that can be compiled and executed in the respective programming environments shifts the focus of the programming language to the complex task of creating algorithms.

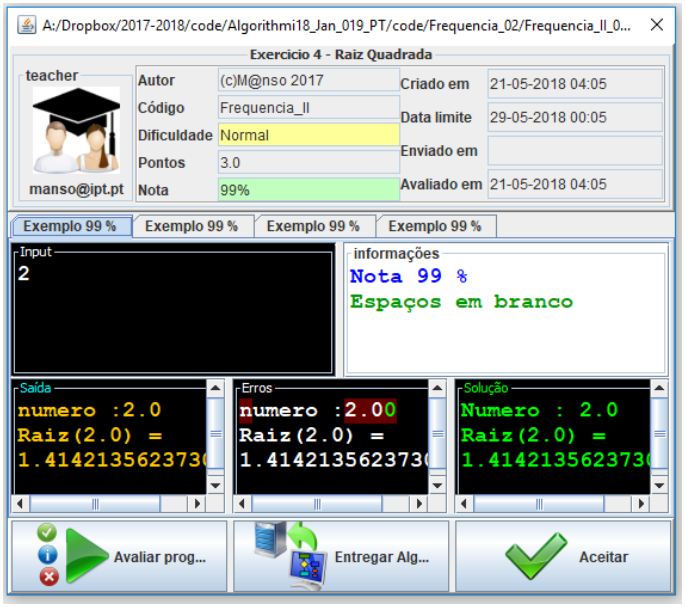


Fig. 8 – Exercise correction

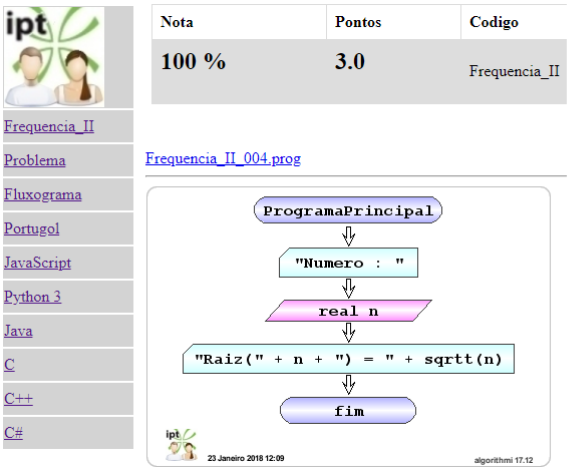


Fig. 9 – Visualization of a solved exercise.

B. *Plagiarism and authentication*

One major problem of teaching programming is plagiarism in exercise-solving. It is very easy to make digital copies and use them as our own. Plagiarism has the negative effect in algorithm learning as students are tempted to obtain the necessary score to pass the course instead of learning algorithmic techniques that can only be acquired with practice.

There is a range of tools that allow the instructors to detect plagiarism and take punitive measures with basis on that information. Punitive measures do not make the learning process more effective and, therefore, Algorithmi uses a different approach.

Students can access algorithms developed by others, study and modify them but cannot copy and use them as their

own. This is achieved through an authentication system in which students' identity is verified.

Authenticated users have access to editor features that are inaccessible to anonymous users. These features also become invisible when the algorithm has not been created by the authenticated user. Thus it is possible to visualize algorithms produced by third-parties but it is not possible to submit them to the information system.

VI. Conclusion

This paper presents Algorithmi, an information system developed at the Instituto Politécnico de Tomar. Algorithmi consists of three modules that articulate to provide an innovative algorithm learning environment. Algorithmic token-based language is amorphous. The editor allows to shape the language and execute it in a configurable environment for the mother tongue of the student; the information system provides the editor with educational content managing student learning.

In addition to the technical features already cited, gamification strategies are also used to increase intrinsic motivation of students through a point system associated to the personal account of each student.

The language extension for the manipulation of complex data and the definition of functions that enable Algorithmi to be used to teach and learn advanced algorithm and data structure concepts.

This system arose from the need to reverse failure rates in introductory programming courses and had the collaboration of the students participating in the assessment of the system usability who validated implemented features and suggested new ones. More advanced-level students also had the opportunity to program some modules of the system.

Algorithmi is at the development stage and its features and usability have been tested during the 2016/2017 2017/2018 academic years. Many corrections have been introduced based on recommendations and detected errors. The assessment of the system in terms of learning outcomes for the CTeSP programme of Information Systems Technologies and Programming and the bachelor's degree in Computer Engineering offered at the IPT is planned for the 2018/2019 academic year.

Bibliographic References:

[1] M. Butler e M. Morgan, "Learning challenges faced by novice programming students studying high level and low feedback concepts", ASCILATE 2007 Singapore, 2007, pp. 99-107.

[2] T. Jenkins, "On the difficulty of learning to program", Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Science, 2002, pp. 27-29.

[3] E. Lahtinen, K. A. Mutka e H. M. Jarvinen, "A Study of the difficulties of novice programmers", Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer ITiCSE'05, 2005, pp. 14-18.

[4] A. Yadin, Reducing the dropout rate in an introductory programming course. ACM Inroads, 2(4), 2011, pp. 71-76.

[5] K. Sloane e M. C. Linn, "Instructional Conditions in Pascal Programming Classes", In R. E. Mayer (Ed.), Teaching and Learning Computer Programming: Multiple Research Perspective. Hillsdale, NJ: Lawrence Erlbaum Associates, 1988, pp. 207-235.

[6] A. J. Gomes, "Dificuldades de aprendizagem de programação de computadores: Contributos para a sua compreensão e resolução", Tese de Doutoramento, Combra: Universidade de Coimbra, 2010.

[7] J. Hidalgo-Céspedes, G. Marín-Raventós e V. Lara-Villagrán, "Learning principles in program visualizations: A systematic literature review", Frontiers in Education Conference (FIE) 2016 IEEE, 2016, pp. 1-9.

[8] A. Manso, C. Marques e P. Dias, "Portugol IDE v3.x: A new environment to teach and learn computer programming", In M. C. Gil, E. T. Caro, M. E. Auer e M. P. B. Merino (Cords.), IEEE EDUCON 2010 Conference - The Future of Global Learning in Engineering Education. Madrid: UPM - Servicio de Publicaciones - EUI – UPM, pp. 1007-1010.

[9] A. C. Nicolodi, "VisualG", Accessed 23 October 2017 at http://visualg3.com.br/

[10] A. Raabe et al., "Portugal Studio", Accessed 14 May 2018 at http://lite.acad.univali.br/portugol/

[11] T. Silva, "G-Portugal", Accessed 8 November 2017 at https://sourceforge.net/projects/gpt.berlios/

[12] H. Abelson, N. Goodman, e L. Rudolph, "LOGO Manual", Massachusetts: Artificial Intelligence Lab, Massachusetts Institute of Technology, 1974.

[13] MIT Media Lab, "Scratch", Accessed 17 November 2017 at https://scratch.mit.edu/

[14] Code.org, "Code Studio", Accessed 17 February 2018 athttps://code.org

[15] A. Manso, L. Oliveira, e C. G. C. Marques, " Ensino da Programação através da Linguagem Algorítmica e Fluxográfica", In C. V. Carvalho, R. Silveira & M. Caeiro (Coord.), TICAI 2009. TIC's para a Aprendizagem da Engenharia. IEEE, Sociedade de Educação: Capítulos Espanhol e Português. Porto: Edições Politema, 2011, pp. 105-110.